

# An Introduction to Variable and Feature Selection

Guyon, Isabelle, and André Elisseeff. "An introduction to variable and feature selection." *Journal of machine learning research* 3.Mar (2003): 1157-1182.

## I. Intro

수천 개부터 많으면 수십 만 개에 달하는 변수를 갖는 데이터가 등장하며 관측값이 충분하지 않음에도 불구하고 irrelevant 하거나 redundant한 변수가 너무 많은 데이터를 처리하는 여러가지 방법이 연구되었다. 위와 같은 문제 상황이 특히 자주 발생하는 도메인은 두 가지가 있는데, 한 곳은 Gene Selection from Microarray Data, 다른 한 곳은 Text Categorization이다.

전자의 경우 변수들은 gene expression coefficients 들인데, 그러한 데이터를 제공하는 환자의 수는 제한적인 것에 반해 채취된 샘플 안의 mRNA의 개수는 굉장히 많기 때문에 large variable situation이 발생한다. 후자의 경우는 주로 문장을 단어 단위로 쪼개고 단어의 등장 빈도 등으로 행렬을 구성하기 때문에 긴 문서일 경우 수백, 수천 개의 변수를 갖는 데이터셋이 만들어질 수 있다.

Variable and feature selection 기법은 여러가지 잠재적인 이점을 가져다준다.

- 데이터 시각화와 탐색을 용이하게 한다.
- 데이터 저장을 용이하게 한다.
- 모델 훈련 과정의 시간을 줄인다.
- 차원의 저주를 피해 예측 성능을 높인다.

따라서 이 paper에서는 여러가지 feature ranking methods와 variable selection methods, 그리고 이러한 방법을 통해 어떻게 더 좋은 성능을 내는 모델을 만들어내는지를 전반적으로 다룬다.

## II. Variable Ranking

그 편리함과 확장성, 그리고 경험적인 사례로부터 유용성이 입증된 variable ranking method는 여러가지 변수 선택 알고리즘 내부에서 적용되고 있다. Large variable situation에서 가장 baseline method로 주로 쓰인다. 이 때 개별 변수들이 종속 변수에 갖는 영향력을 기준으로 rank를 매기기 때문에 marginal한 방법이라고 볼 수 있다. Filter method라고도 자주 불리며, predictor의 선택과 정과는 독립적으로 이루어진다.

### II.I. Correlation Criteria

연속적인 종속변수  $y$ 가 존재하는 경우를 상정해보자. 피어슨 상관계수는 다음과 같이 정의된다.

$$\mathcal{R}(i) = \frac{\text{cov}(X_i, Y)}{\sqrt{\text{var}(X_i)\text{var}(Y)}}$$

이  $\mathcal{R}(i)$ 의 estimator는 아래의 식으로부터 구할 수 있다.

$$R(i) = \frac{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)}{\sqrt{\sum_{k=1}^m (x_{k,i} - \bar{x}_i)^2 \sum_{k=1}^m (y_k - \bar{y})^2}}$$

선형 회귀에서 결정계수는  $R(i)^2$ 이기 때문에, 개별 변수들에 대한 선형 회귀 적합도 검정의 연장선으로서 결정계수를 variable ranking의 기준으로 사용하는 것의 타당성이 부여된다. 그러나 이러한 피어슨 상관계수로부터의  $R(i)^2$ 은 변수간의 선형적 의존성만을 가지고 rank를 매기기 때문에, 이러한 제약을 타파하기 위해 개별변수와 종속변수 간의 비선형 모델을 만들어 적합도 검정을 실시하거나, 이러한 과정에서 과적합이 우려된다면 비선형 관계의 데이터를 적절하게 transform(log, inverse, etc.)하여 simple correlation coefficient를 사용할 수도 있다.

### II.II. Single Variable Classifiers

Regression이 아닌 Classification problem에서는 어떻게 variable ranking을 구현할 수 있을까? 개별 변수와 종속변수 간의 classifier model을 세우고 개별 classifier의 성능을 기준으로 rank를 매길 수 있다. 단순히 error rate를 기준으로 performance를 평가할 수도 있지만 fpr과 fnr를 고려해 ROC Curve의 형태나 AUC를 기준으로 ranking을 진행할 수 있다.

## II.III. Information Theoretic Ranking Criteria

정보 이론에 기반하여 변수들의 등수를 매기려는 시도 또한 등장했다. (Bekkerman et al., 2003, Dhillon et al., 2003, etc.)

이와 관련된 대부분의 연구는 개별 변수와 종속 변수 사이의 상호 정보량에 대한 경험적인 추정치에 의존한다.

$$\mathcal{I}(i) = \int_{x_i} \int_y p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} dx dy$$

$p(x_i)$ 와  $p(y)$ 는  $x_i$ 와  $y$ 의 probability density이고  $p(x_i, y)$ 는 joint density이다.

위 measure의 가장 큰 단점은 그러한 probability densities와 joint density를 알기 어렵다는 점이다. 데이터로부터 추정해내기도 쉽지 않다. 이산형이나 명목형 변수의 경우에는 integral term이 summation으로 바뀌고 probability mass를 빈도수를 통해 결정 지을 수 있기 때문에 비교적 정보량의 추정치를 구하기 쉽다.

$$I(i) = \sum_{x_i} \sum_y P(X = x_i, Y = y) \log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)}$$

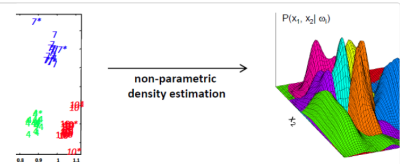
그러나 연속형 변수의 경우에는 이러한 추정치를 얻기가 매우 어렵다. 따라서 이러한 경우에는 변수를 이산화 시켜버리거나 Parzen windows 같은 방법으로 non-parametric하게 확률밀도를 추정하는 방식을 택할 수 있다.

### ▼ Parzen Window 추가 설명 (a.k.a KDE)

#### 커널 밀도 추정(Kernel density estimation) - Parzen window density estimation

다른 밀도 추정법들이 데이터가 특정 분포를 따른다는 가정 하에 추정하는 방법이었습니. 하지만 이번에 설명할 커널 밀도 추정은 데이터가 특정 분포를 따르지 않는다는 가정 하에 밀도를 추정하는 방법입니다. 커널 밀도 추정의 기본적인 개념을 알아보고 대표적인 그 중 파젠 윈도우 밀도 추정(Parzen window density

[https://jayhey.github.io/novelty%20detection/2017/11/08/Novelty\\_detection\\_Kernel/](https://jayhey.github.io/novelty%20detection/2017/11/08/Novelty_detection_Kernel/)



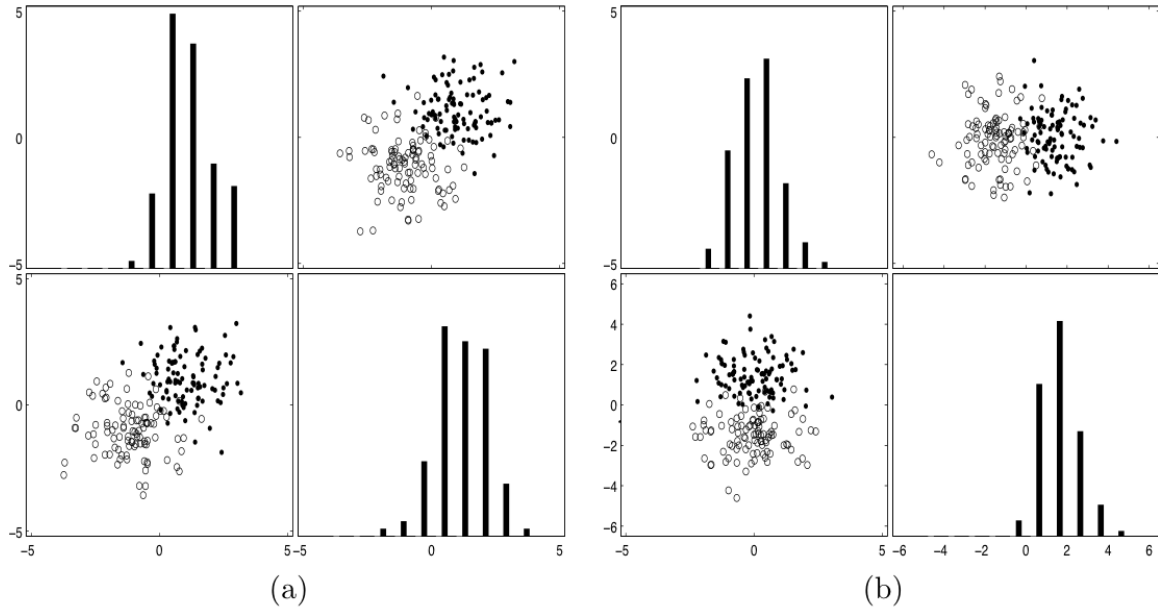
만약 정규 분포를 이용해 확률밀도를 추정한다면  $X_i$ 와  $Y$ 의 covariance를 구하는 것과 같아지므로 correlation coefficient를 이용한 ranking 방법으로 회귀하게 된다.

## III. Small but Revealing Examples

세 번째 섹션에서는 간단한 example을 통해 variable ranking method에 대한 여러가지 의문에 대한 대답을 제공하고, 그와 동시에 이 방법이 필연적으로 갖는 한계에 대해 조명한다.

### III.I. 불필요한 변수들까지 포함하는 게 성능 향상에 도움이 될까?

Variable ranking method에 대해 가장 흔하게 들어오는 태클은 랭킹으로부터 도출한 변수들의 subset이 지나치게 크다는 점이다. 그보다 훨씬 작은 subset만으로도 같은 성능에 충분히 도달할 수 있기 때문이다. 저자는 간단한 예시를 들어서 presumably redundant한 변수들도 성능 향상에 도움을 줄 여지가 있음을 보인다.



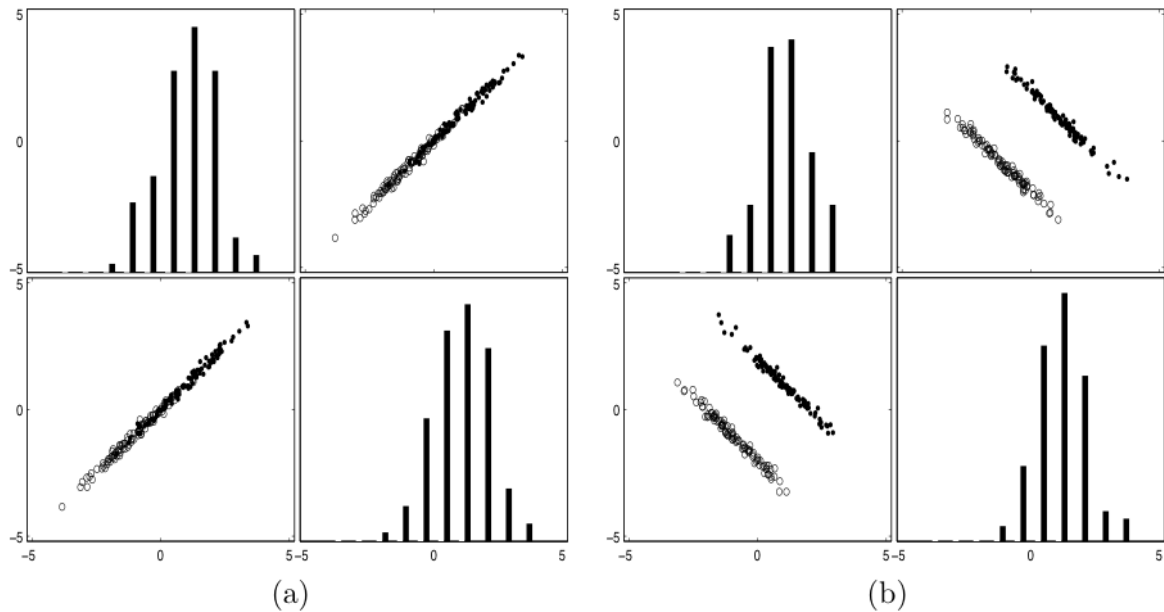
출처: Guyon et al., 2003

위 그림에서 두 클래스의 중심은 각각  $(-1, -1)$ ,  $(1, 1)$ 에 위치하고 있고, 두 변수들은 표준편차 1을 갖는 정규분포로부터 iid하게 추출되었다. 좌측 그림에서 두 클래스의 중심을 x축 방향으로 project 했을 때 그 거리는 2가 된다. 우측 그림은 동일한 관측값을 45도 회전시킨 후 scatter한 그림인데, 이 때 두 클래스의 중심을 x축 방향으로 사영하면 그 거리가  $2\sqrt{2}$ 로, 이전보다 더 잘 separate됨을 알 수 있다.

따라서 어떤 확률분포로부터 IID하게 추출된 샘플로 이루어진 변수들은 redundant하게 보일지라도 실제로는 redundant하지 않을 수 있고, 오히려 classification 등의 task에 도움이 될 수 있다.

### III.II. 변수간의 상관관계가 변수의 유용함에 미치는 영향

변수간의 상관관계는 모델 성능에 해당 변수들이 갖는 유용함을 따질 때 중요하게 고려해야 할 요소이다.



출처: Guyon et al., 2003

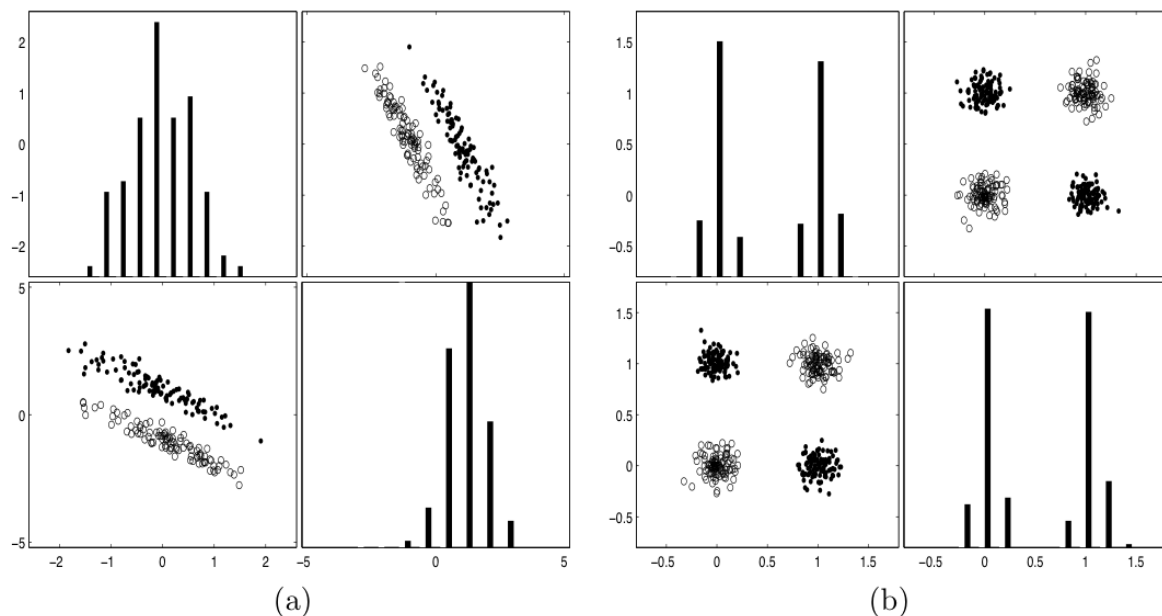
위 그림은 이전과 동일한 class center를 갖는 데이터들에 대한 plot인데, 이번에는 두 클래스의 중심을 관통하는 방향으로 분산이 크고 그와 직교하는 방향으로 분산이 매우 작은 형태이다.

이러한 형태를 perfectly correlated 되었다 라고 하는데, 그림으로부터 알 수 있듯이 아무리 plot을 rotate해도 유용한 separation point를 찾을 수 없음을 쉽게 깨달을 수 있다. 즉, **perfectly correlated된 변수들은 truly redundant 하다**. 두 변수를 추가함으로써 얻을 수 있는 정보가 더 이상 없기 때문이다. 그러나 오른쪽 figure에서는 first principal direction이 클래스 중심과 직교하는 방향이다. 즉, 두 변수가 anti-correlation을 갖는다. 이러한 형태에서는 그림으로부터 알 수 있듯이 정확한 separation이 가능하다. 이로부터 알 수 있는 사실은 **highly correlated variable** 이라고 해서 서로 complement할 여지가 아예 없는 건 아니라는 점이다.

III.I과 III.II로부터 알 수 있는 사실은 개별적으로는 presumably redundant 하더라도 모델에는 유용하게 쓰일 '수도 있다'는 것이다. 그러나 이미 앞듯이 variable ranking method는 아주 marginal한 성격을 갖고 있기 때문에 어떠한 형태의 combination이 모델 성능에 가장 최적인지를 판단하기에는 어렵다는 치명적인 단점이 존재할 수밖에 없다.

### III.III. 완전히 쓸모 없는 변수도 함께 쓰이면 쓸모가 있다?

위에서 언급했듯이 marginal한 approach에는 분명한 단점이 존재한다. 그렇다고 multivariate method를 쓰자니 과적합이 걱정된다. 그렇기 때문에 filtering method를 먼저 적용하고 남은 변수들만 가지고 multivariate하게 고려를 시도하는 방향으로 연구가 이루어지는 것 같다.



출처: Guyon et al., 2003

위 그림은 completely useless variable도 다른 variable과 함께 고려되었을 때 유용하게 쓰일 수 있는 여지가 존재함을 보여주는 예시들이다. 좌측의 경우 x축과 y축 모두 유용한 separation을 제공하지 못하지만, 어쨌든 둘이 함께 쓰였을 때는 2차원 평면상에서 유의한 구분이 가능해진다. 우측은 XOR problem의 경우인데, 마찬가지로 x축과 y축 모두 linear한 separation에는 실패할 수밖에 없지만 2차원 평면상에서 함께 쓰였을 때는 class의 구분이 가능해진다. (역시 linear하게는 아니지만)

섹션 3의 내용을 고려해 사건을 덧붙이자면, 결국 ranking method는 관측값의 개수에 비해 변수의 개수가 지나칠 정도로 많은 상황에서만 사용해야 할 것 같다. 실제로 통계적 모델링과 머신러닝 강의에서도 김재직 교수님이  $p > n$  상황에서만 고려하라고 말씀하시기도 했다. 또, ranking method를 쓰게 된다면 한가지 criterion만 사용하기보다 여러가지 criteria를 종합적으로 고려해 voting을 통해 버릴 변수를 선택한다든가, 아니면 단순히 linear한 관계만 따지기보다 각 변수별로 정확한 non-linearity까지 따져서 rank를 매긴다거나 (근데 이건 불가능할듯) 하는 식으로 최대한 정교하게 버릴 변수를 선택해야 할 것 같다.

## IV. Variable Subset Selection

그렇다면 함께 고려될 때 유용한 변수들을 살리기 위해 variable ranking 대신 어떤 방법을 사용할 수 있을까? 그렇게 고안된 것이 바로 variable subset selection method이다. 이 방법은 크게 세 가지로 나뉜다.

- Wrappers: 학습 모델 자체는 블랙박스로 두고 오로지 예측 성능에 따라서만 변수 조합에 점수를 매긴다.
- Filters: 모델과 독립적으로 학습단계 이전에 변수들의 하위 집합을 결정한다.
- Embedded: 학습 과정 중에 변수 선택을 함께 수행한다.

## IV.I. Wrappers and Embedded Methods

Wrapper method는 간단하면서도 강력한 변수 선택 방법을 제시한다. 학습 모델에 관계 없이 그저 가장 좋은 예측 성능을 보이는 변수들의 하위집합을 찾아내는 과정이기 때문이다. 변수의 개수가 지나치게 많은 게 아니라면 전역적인 search 역시 이론상 가능하지만, 어쨌든 적절한 subset을 찾는 것은 NP-hard 문제이기 때문에 계산이 절대 용이하지 않다. Best-first, branch-and-bound, simulated annealing 등 다양한 탐색 전략이 존재하며, 점수의 기준으로 쓰이는 예측 성능을 계산할 때는 주로 validation set을 따로 사용하거나 교차 검증을 이용한다.

Wrappers method가 흔히 비판받는 부분은 지나치게 많은 계산을 필요로 한다는 이유 때문인데, 실상은 그렇지 않다. 여러가지 효율적인 탐색 전략들이 이미 존재하고, 그것들을 사용한다고 해서 모델 성능이 악화되지도 않는다. 오히려 조금 투박한(= 완벽하게 정교하지 않은) 탐색 전략일수록 과적합에 더 강건하다고 한다. 또한 greedy search algorithm은 계산적으로도 감당할만하고 또한 과적합에도 예민하지 않다. Forward selection이나 backward elimination이 이러한 탐욕적인 탐색 알고리즘을 가진 wrappers method에 속한다.

Wrappers method가 비록 고루 쓰이고 또 간단하지만, 몇몇 측면에서는 모델 훈련 과정 속에 변수선택 단계가 통합되어 있는 Embedded method가 더욱 효율적일 수 있다. 데이터를 따로 train과 validation으로 나눌 필요도 없고, 또한 여러가지 변수 조합에 따라 모델을 재훈련시킬 필요도 없기 때문이다. 그렇다면 Embedded method에서는 어떤 것을 기준으로 변수를 선택할까?

## IV.II. Nested Subset Methods

Embedded method는 변수 조합을 바꿔가며 그 때마다 변화하는 목적함수를 기준으로 최적의 조합을 탐색해나간다.  $s$ 를 선택된 변수의 개수라 하고  $J(s)$ 를 해당 변수들을 가지고 얻은 목적함수의 값이라고 할 때 아래 세 가지 기준에 따라 목적함수의 변화량을 측정한다.

- **유한차분:**  $J(s)$ 와  $J(s+1)$  또는  $J(s-1)$ 과의 차이를 계산한다.
- **손실함수의 이차근사:** 주로 후진소거법에 쓰이며, 독립변수의 가중치를 없애나가는 방향으로 이루어진다. 테일러 전개를 통해 이차근사식을 얻어낼 수 있으며, 그것의 변화량을  $DJ_i = (\frac{1}{2}) \frac{\partial^2 J}{\partial w_i^2} (Dw_i)^2$ 로 표현할 수 있다. 이에 따라  $i$ 번째 변수를 소거할 때의 변화량이  $Dw_i$ 의 변화를 통해 얻어진다.
- **목적함수의 Sensitivity:**  $J$ 를  $x_i$ 에 대해 미분한 값의 절대값이나 제곱근의 값으로 sensitivity를 계산한다.

먼저 유한차분 방법의 경우는 그 계산 과정이 효율적이기 때문에 여러가지 훈련 알고리즘에서 사용된다. 그람-슈미트 직교화 과정에서 MSE의 변화량을 근거로 변수를 추가하는 방식으로 모델이 작동하는데, 이 개념도 유한차분의 개념이다. 이러한 선형 모델 말고도 비선형 모델에서도  $J$ 의 유한차분을 근거로 변수를 선택할 수 있는데, 특히 kernel method에서는 커널 함수의 계수인  $\alpha_k$ 를 고정하는 식으로  $J$ 의 유한차분을 효율적으로 근사할 수 있기 때문에 SVM 등의 모델에서 변수 선택할 때 이 개념이 쓰인다.

손실함수의 이차근사 방법은 'optimum brain damage(OBD)'라고도 불린다. Rivals and Personnaz (2003)에 따르면  $DJ_i$ 를 사용하는 것이  $|w_i|$ 를 사용하는 것보다 변수 가중치를 소거하는 기준으로 더 적합하다고 한다. 그러나  $J$ 가  $\sum(\mathbf{w} \cdot \mathbf{x}_k + b - y_k)^2$  꼴로 정의되는 선형차승모델이나  $\frac{1}{2} \|w\|^2$ 를 minimize 해야 하는 선형 SVM의 경우 등,  $J$ 가  $w_i$ 의 이차항을 포함하는 경우는  $DJ_i$ 를 쓰나  $|w_i|$ 를 쓰나 결과가 같다.

## IV.III. 모든 것을 위한 Objective function

모델링에서 변수 선택을 직관적으로 이해해보면, 변수 선택을 위한 Objective function은 goodness-of-fit은 maximize하고 number of variables는 minimize 하는 식이 되어야 한다. 이러한 접근은 goodness-of-fit과 regularization term을 동시에 다루는 two-part objective function과 그 궤를 같이 한다. Weston et al. (2003)에 의해 처음 제시된 이 아이디어는 linear한 SVM에 적용되어 설명되는데, 그들은 vanilla SVM을 약간 변형하여  $l_0$ -norm SVM의 해를 근사적으로 찾을 수 있음을 보였다.

1. Linear SVM을  $l_1$ -norm 또는  $l_2$ -norm을 사용해 훈련시킨다.
2. 1단계에서 얻은 모델의 가중치  $\mathbf{w}$ 의 절대값을 곱해줌으로써 독립변수를 re-scale 한다.
3. 결과가 수렴할 때까지 위 두 단계를 반복한다.

마치  $|w_i|$ 의 크기에 따라 작은 것부터 소거하는 backward elimination과 비슷한 과정이다. ( $l_1$ 이나  $l_2$  규제를 사용하면 가중치가 0이 되어 변수가 사라지거나 매우 0에 가까운 가중치를 갖는 변수들이 생길것이므로 그것들을 제거하는 느낌인듯?)

이에 더해서 Perkins et al. (2003)에서는 규제를 통한 변수 개수 컨트롤 후 남게 되는 변수의 개수에 대한 규제까지 더하여 three-part objective function을 제시했다.

그럼에도 불구하고 non-linear한 모델에 대해서 직접적으로 변수의 개수를 최소화하는 objective function은 없다. 그 대신, 몇몇 학자들은 non-linear model에 대한 변수 선택 문제를 변수 스케일링의 문제로 치환했는데, 이 때 그 scaling factor는 최적값을 찾아야 하는 하이퍼 파라미터이다.

## IV.IV. Filters?

Wrapper, Embedded methods와 Filter methods의 가장 큰 차이는 subset selection 과정에서 모델 학습이 필요한가 그렇지 않은가의 차이이다. 그렇기 때문에 filter는 훨씬 빠르다. 또한 Filters가 다른 방법들보다 모델에 구애받지 않는 일반적인 subset을 찾아준다는 의견도 있다. 그러나 Filters의 가장 큰 쓰임은 바로 차원을 줄이고 과적합을 방지하기 위한 방법으로 전처리 과정으로서 사용될 수 있다는 점이다.

이러한 관점에서 선형 모델로 Wrapper나 Embedded method를 구현해서 변수를 줄이고, 이후에 남은 변수들만을 가지고 task를 수행할 때는 non-linear한 모델을 사용하는 방법들이 등장한다. (왜 굳이 linear하게 wrapper나 embedded를 써야하는지는 모르겠다.) 예를 들어서 Bi et al. (2003)에서는 linear  $l_1$ -norm SVM으로 variable selection을 진행하고 non-linear  $l_1$ -norm SVM은 예측에 사용했다고 한다. 이해한 바로는 Filters는 그에 속하는 정확한 개념이 있다기보다, pre-processing의 일부로 variable selection을 진행하는 경우들을 통칭하는 것 같다.

## V. Feature Construction and Space Dimensionality Reduction

지금까지 논의한 것처럼 변수 선택을 통한 차원의 축소는 여러가지 측면에서 시간과 속도 등을 단축시켜주기 때문에 이득이 된다. 그러나 시간이나 속도의 절약을 필요 없는 상황이라면 차원 축소를 하기 위해 반드시 변수 선택을 활용할 필요는 없다. 머신러닝의 정수는 데이터의 적절한 representation을 만들어내는 것에 있다. Original input으로부터 변수를 새로 조합함으로써 좋은 성능을 기대할 수도 있는 것이다. 특히 이러한 작업을 통해 우리는 해결하려는 문제에 대한 도메인 지식과 데이터 및 분석 기술의 통합을 이끌어낼 수 있다. 좋은 말 같아서 bold처리

변수 선택 대신 기존의 변수를 적절하게 조합하거나 혹은 기타 다른 방법으로 차원을 줄이는 방법을 feature construction이라고 명명하는 것 같다 (변수 개발이라고 부르겠다). 이러한 변수 개발의 목표는 두 가지로 나뉜다: 1. 위에서 말한 것처럼 데이터의 best representation을 만들어내거나 2. 이를 통한 예측 과정을 가장 효율적으로 만든다. 전자의 경우 비지도적인 방법으로 달성할 수 있으며, 데이터 압축의 영역에서 역시 쓰이는 기법이라고 한다. 후자의 경우는 지도적인 방법으로 달성할 수 있는 목표이다. 그렇다면 지도 학습에서 비지도적인 방법으로 변수 개발을 할 필요가 있을까? 논문에서는 비지도적인 방법으로 개발한 변수가 과적합에 적게 민감하기 때문에 시도할 필요가 있다고 언급한다. 이 섹션에서는 클러스터링과 MF를 통한 변수개발, 그리고 이보다 더 지도적인 방법을 통한 변수개발 방법을 설명한다.

## V.I. Clustering

클러스터링은 변수 개발의 영역에서 오랫동안 쓰여왔다. 당연히 그 개괄적인 개념은 비슷한 변수들을 클러스터로 묶어 클러스터의 중심으로 대표되는 새로운 변수로 만드는 것이다. 클러스터링은 비지도 학습이지만, 클러스터간 더 차별적인 차이를 만들기 위해 지도적인 방법이 그 과정에서 쓰이는데, 이를 distributional clustering이라고 한다. 이 방법은 Information Bottleneck 이론으로부터 출발한다.

$\tilde{X}$ 를 새로 개발된 변수라고 한다면, IB 이론은 기존 변수와 개발된 변수 간의 상호 정보량  $I(X, \tilde{X})$ 는 최소화하고 종속 변수와 개발된 변수 간의 상호 정보량  $I(\tilde{X}, Y)$ 는 최대화 하는 방향을 지향한다. 따라서 이 목적을 달성하려는 목적 함수는 라그랑주 승수를 접목시켜  $J = I(X, \tilde{X}) - \beta I(\tilde{X}, Y)$ 의 꼴로 표현할 수 있게 된다.

자연어 처리 분야는 클러스터링을 통한 변수 개발이 자주 쓰이는 분야이다. BoW를 생각해 보면 각 변수는 해당 단어가 각각의 문서에서 얼마나 등장하는지에 대한 횟수를 담고 있다. 그러므로 당연히 각 변수들을 묶어서 그 수를 줄이는 방향으로 작동하며, 이 때 각 문서들(즉, each row)은 IB에 따라 각 문서의 카테고리별로 묶여 재분류된다. 따라서 BoW가 해당 단어가 각 문서에서 얼마나 등장하는지를 알려주는 것에서 각 단어의 클러스터가 각 문서의 카테고리별로 얼마나 등장하는지 알려주는 형태로 변화하는 것이다.

가장 심플한 적용 방법은 Dhillon et al. (2003)에 의해 제시되었으며, 이 때 기존 단어 변수와 단어 클러스터 변수의 similarity는 K-L Divergence를 사용해  $K(\mathbf{x}_j, \tilde{\mathbf{x}}_i) = \exp(-\beta \sum_k x_{k,j} \ln(\frac{x_{k,j}}{\tilde{x}_{k,i}}))$ 로 구해진다. K-L Divergence에 대해서는 [참고](#) 확인. 시그마 합에서  $k$ 는 각 문서의 카테고리를 의미하기 때문에 해석해보자면 각 카테고리별로 기존 변수와 클러스터링으로 묶인 변수들간의 크로스 엔트로피의 차이를 의미한다. 즉, 값이 클수록 두 변수들의 확률 분포에서 차이가 존재한다는 뜻 (= 개발된 변수들이 기존의 변수와 차별되었다는 뜻)이 된다. **(확인필요)**

조금 더 정교한 방법으로는 Bekkerman et al. (2003)에 의해 제시된 소프트한 버전의 K-means 방법이 있는데, 기존의 변수들(각각의 단어)이 여러가지 클러스터에 동시에 포함될 수 있도록 허용하고 라그랑주 승수인  $\beta$ 를 조정함으로써 클러스터를 점진적으로 분할하는 방식이다. 두 가지 방식 모두 잘 작동하며, 다만 Bekkerman의 방식에서 여러가지 클러스터에 동시에 포함되는 단어들이 거의 없는 것으로 보아 굳이 soft한 방식을 쓰지 않아도 충분하다는 점이 시사된다.

#### ▼ 참고

K-L Divergence, 즉 쿨백-라이블러 발산(차이)은 두 확률분포를 비교할 때 사용한다. 그 수식을 보면

$$D_{KL}(P|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

로 나타낼 수 있는데, 이를 해석해보자면

$P$ 는 실제 확률분포,  $Q$ 는 가설의 확률분포이다. 위 수식을 분해해보면

$$D_{KL}(P|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} = \sum_i P(i) \log P(i) - \sum_i P(i) \log Q(i) = -H(P) + H(P, Q)$$

즉,  $P$ 와  $Q$ 의 크로스 엔트로피로부터  $P$ 의 엔트로피를 뺀 꼴이 되므로, 결과적으로  $Q$ 가  $P$ 를 얼마나 잘 흉내내느냐, 즉  $P$ 를  $Q$ 로 근사했을 때 정보량이 얼마나 차이가 나는가를 측정하는 도구인 셈이다.

## V.II. Matrix Factorization

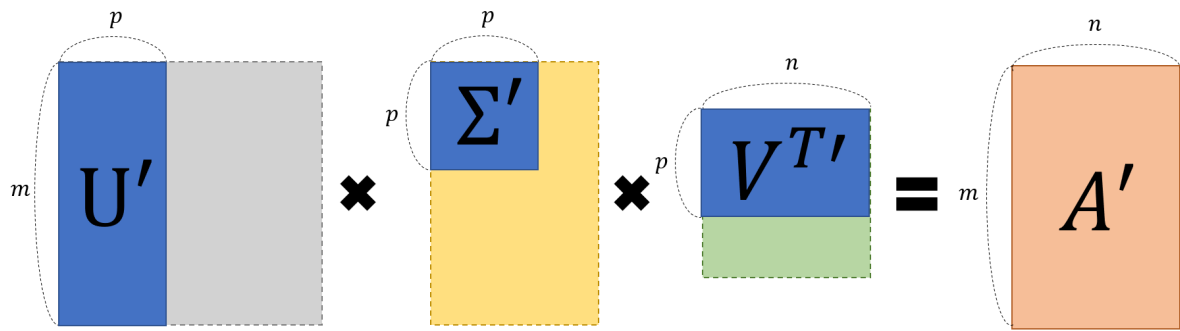
SVD를 이용한 MF 방식도 비지도적인 방법 중 일부러 변수 개발 할 때 쓰인다 (고한다). Globerson and Tishby(2003)는 정보 이론에 기반한 비지도적인 변수 개발 방법인 Sufficient Dimensionality Reduction (SDR)을 제시했다. 가장 정보량이 많은 변수는 Data Reconstruction과 Data Compression 사이의 tradeoff를 다루는 최적화 문제를 해결함으로써 얻어낼 수 있다는 개념,,, 이다.

두 가지 확률 변수의 결합 확률 분포를 나타내는  $m \times n$  Non-negative matrix가 있을 때, 정보 이론의 정보투영을 이용해 feature를 추출하여 새롭게 구조화된 행렬  $\tilde{P} = (\frac{1}{Z}) \exp(\Phi \Psi)$ 을 만들어낸다. 이에 대한 논문은 아래 참고 확인.

#### ▼ 참고

<https://www.jmlr.org/papers/volume3/globerson03a/globerson03a.pdf>

사건을 덧붙이자면 SDR 방법을 정확히 이해하는 것보다 SVD를 활용한 변수 개발을 직관적으로 이해하는 게 더 중요해보인다. SVD는  $A = U \Sigma V^T$ 으로 분해하는 것이고, 그렇게 분해하는 이유는  $U \Sigma V^T$ 의 일부 행과 열만을 가지고도  $A$ 와 근사한 행렬  $A'$ 을 만들어낼 수 있기 때문이다.



이 때 위에서 언급한 Data Reconstruction과 Data Compression 사이의 tradeoff라고 한다면, Data Reconstruction은  $A$ 와  $A'$  사이의 최소차승으로 계산한 차이일테고, Data Compression은  $XV = U\Sigma$ 에서  $XV$ 를 통해 달성할 수 있는 PCA에서 차원이 얼마나 축소되도록 할 건지를 의미할 것이다. 즉,  $A$ 와  $A'$ 의 reconstruction error는 최소화 하면서 최소한의 PC로 차원을 축소하는 optimization 문제를 해결하면, 그 때 선택되는 PC들이 가장 정보량이 많은 변수들이 되는 것 아닐까? 어렵다..

### V.III. Supervised Feature Selection

마지막으로는 지도적인 방법으로 변수 개발을 달성하는 방식들인데, 음,, general 하기 보다는 그냥 특정 논문들에서 제시된 방법들이다. MF와 마찬가지로 개념적 쓸모를 이해하는 데만 집중하면 좋을듯 하다.

- **Nested Subset Methods**

위에서 이미 언급한  $DJ_i$ 의 변화량을 통해 특정 변수의 가중치를 없애는 방식인데, 이것이 왜 feature construction으로 불리느냐 하면, 바로 neural network에서 특정 internal node 들을 pruning 하는 과정에서 이 개념이 쓰이기 때문이다. 그람-슈미트 직교화가 OBD의 대체 방식으로 제시되기도 한다. (위에서 설명한 것에서 손실함수의 이차 근사 대신 유한 차분을 써도 되는 것처럼)

- **Filters**

Torkkola (2003)에 따르면 상호 정보량을 기준으로 변수를 개발하는 방법으로 filter method가 쓰일 수도 있다.

- **모든 것을 위한 objective function**

위에서 언급했듯이 변수 선택의 과정은 two-part optimization으로 해석할 수 있는데, 만약 linear한 form이 아닐 경우 그런 꼴로 변수 선택을 최적화처럼 풀어내는 것이 힘들다고 했다. 이에 Weston et al. (2003)에서는 Kernel을 통해 고차원으로 데이터를 치환하는 변수 개발 방법을 제시함으로써  $l_0$ -norm 을 최소화하는 objective function을 보이기도 했다.

## VI. Validation Methods

이 섹션에서는 변수 선택과 그 결과로 탄생한 모델에 대한 validity를 검증하는 방법에 대해 설명한다. 수리적인 내용보다, 앞선 섹션과 마찬가지로 그 개념적인 이해를 시도하는 게 중요해보인다.

모델의 performance를 평가하기 전에 당연히 어떤 모델을 사용할 것인지를 결정해야 한다. 이를 Model selection이라 하는데, 이 과정에서 feature selection, variable selection, hyperparameter tuning 등이 모두 이루어진다. 여러 모델들 중에서 가장 좋은 모델을 고르기 위해 데이터는 먼저 training과 test로 나뉘어야 하고 training은 다시 train과 validation으로 나뉘어져야 한다. 당연히 fixed validation set만을 만들 수도 있고 여러가지 cross validation을 사용할 수도 있다.


Validation error를 기준으로 모델을 선택하게 될텐데, 이 때 제기되는 문제가 바로 모델들 간의 validation error의 차이를 어떻게 받아들일 것인가 하는 문제이다. Fixed validation set의 경우에는 여러가지 통계적 가설검정을 시도할 수 있다. 하지만 Cross validation이 사용되었을 경우 statistical tests의 liability가 훼손될 수밖에 없는데, 바로 샘플들 사이의 iid 가정이 깨지기 때문이다. 이에 대해 여러 학자들의 의견과 방법이 제시되었다. 그런데 sample이 매우 많다면 굳이 데이터셋을 여러 개로 나눌 필요 없이 fixed validation set만 가지고 통계적 가설검정으로 모델들끼리 비교해도 괜찮다고 한다. 아니면 iid 가정이 필요없는 시계열 데이터의 경우는 cross validation을 쓰는게 유효하고, 또 CV를 쓰는 경우는 에러끼리의 가설검정보다는 confidence interval을 비교하는 식으로 model selection을 진행할 수도 있다. 이에 대한 내용은 잘 정리된 문서가 하나 있으니 아래 참고

- ▼ **참고**



#### Statistical Significance Tests for Comparing Machine Learning Algorithms - Machine Learning Mastery

Comparing machine learning methods and selecting a final model is a common operation in applied machine learning. Models are commonly evaluated using resampling methods like k-fold cross-validation from which mean skill scores are calculated and compared directly.

 <https://machinelearningmastery.com/statistical-significance-tests-for-comparing-machine-learning-algorithms/>



데이터의 어느 정도를 train으로 쓰고 어느 정도를 validation으로 쓸지에 대한 논의에는 정답이 없다. 그런데 많은 연구자들이 high variance의 위험에도 불구하고 LOOCV를 권장한다고 한다. (왜인지 모르겠음,,, 논문 읽어보기 \_ 참고)

#### ▼ 참고

Monari and Dreyfus, 2000

Rivals and Personnaz, 2003

Rakotomamonjy, 2003

LOO error는 training error의 교정된 값이라고 볼 수 있다. 정확한 해석은 없지만 대충 느낌대로 가보자면 LOO가 sample 한 개 빼고 나머지  $n-1$ 개로 학습하고, 그 빼놓은 sample로 테스트 하는 식인데, 모든 데이터가 한 번씩은 sample이 되니까, 정확한 training error의 penalized 된 버전이라고 보는데? 이렇게 training error에 penalization을 주는 식의 validation error 비교 방법을 metric-based methods라고 부르는데, Bengio and Chapados (2003)은 이러한 방법을 변수 선택에 적용하는 개념을 제시했다.

두 가지 모델  $f_A$ 와  $f_B$ 가 있을 때,  $A$ 는  $B$ 라는 변수 집합의 nested subset이다. 즉,  $A \subset B$ 인 셈. 이 때 두 모델의 차이를  $d(f_A, f_B)$ 라고 표현할 때, Bengio and Chapados (2003)이 제시한 criterion은  $d_U(f_A, f_B)/d_T(f_A, f_B)$ 이다. 분자  $d_U(f_A, f_B)$ 는 unlabelled data로 계산한 discrepancy이고, 분모는 training data로 계산한 discrepancy이다. 즉 criterion의 값이 크면 클수록 전체 변수를 사용한 모델과 그 변수들의 nested subset을 사용한 모델의 성능 차가 있다는 뜻이고, 따라서 변수 집합  $B$  중에서  $A$ 에 쓰이지 않은 새로운 변수를 추가해 볼 여지가 생기는 것이다.

Variable ranking 과정에서 적용될 수 있는 독특한 방법 하나 역시 소개가 되었는데, 바로 새로운 확률변수를 변수로서 추가하는 방법이다. 아주 쉬운 응용 예시로서 Bi et al. (2003)에서는 기존의 true variable 집합에다가 “fake variables” 3개를 추가했다. 이들은 정규분포를 따르는 샘플들에서 random하게 추출되어 true variables들과 함께 variable ranking 과정에 관여하게 된다. 따라서 이 때 각 변수가 갖는 relevance가 fake variable들보다 낮거나 혹은 비슷한 수준이라면 삭제해도 괜찮은 변수라는 결론을 도출할 수 있다. 조금 더 정교하게 fake variable을 만들고 싶다면 기본적으로 가우시안 분포를 따르는 random variable을 만들고 거기다가 확률적으로 true variable의 vector 일부를 섞는 방식으로 만들 수도 있다.

## VII. Advanced Discussions

### • 변수선택 과정의 분산을 줄이는 방법

전체 주제에 대해서 추가적으로 몇 가지 이야기들을 묶어서 정리하려 한다. 변수 선택의 과정은 작은 변화에도 민감하게 반응한다. 알고리즘의 초기 설정, 몇 가지 변수의 삭제나 추가, 아니면 샘플의 노이즈 등등 작은 동요에 따라 비슷한 예측력을 보이는 서로 다른 subset들이 여러 개 등장할 수도 있다. 즉 변수 선택의 과정은 분산이 높다는 뜻인데 머신러닝에서 분산이 높은 건 당연히 안 좋다. 그래서 변수 선택 과정을 stabilize 하는 방법들에 대한 연구가 많이 진행되었는데, 가장 좋은 방법은 역시 부트스트랩이다.

Training data로부터 부트스트랩 된 sub sample들을 가지고 변수 선택을 반복적으로 수행하고, 이 때 중복으로 등장하는 변수들이 있다면 이들은 ‘stable’한 변수라고 볼 수 있을 것이다. 또한 부트스트랩의 장점은 선택되는 변수들을 가지고 등수를 매길 수 있다는 점인데, 여러 샘플에서 자주 등장하는 변수일수록 더욱 relevant할 가능성이 커지기 때문이다. 이런식으로 variable ranking을 한다면 논문의 초반에 ranking method가 marginal한 영향력만 측정할 수 없던 것과 다르게 다른 변수들과의 관계를 고려한 ranking도 가능해진다.

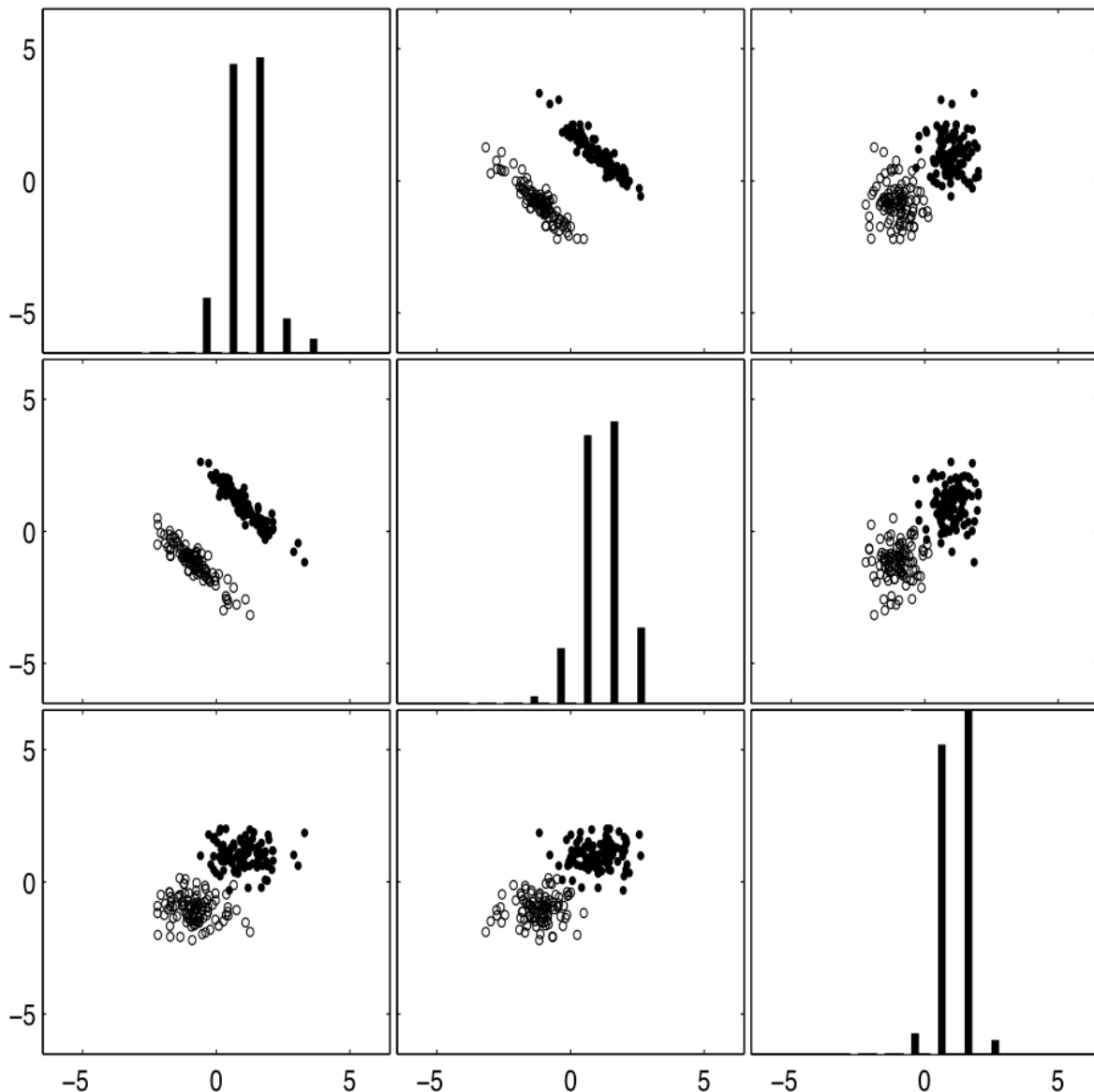
이렇게 다른 변수와의 맥락을 고려한 variable ranking의 다른 방법으로는 Relief 알고리즘이 있다. 통계적 모델링과 머신러닝 시간에도 배운 알고리즘인데, 기본적으로 binary classification 문제에서만 작동 가능하다.(multiple category로 확장할 수도 있음) 우선 랜덤한 훈련 샘플  $x_i$ 를 선택하고, 그 샘플과 동일한 라벨에 속하는 가장 가까운 다른 훈련 샘플  $x_H$ 와 다른 라벨에 속하는 가장 가까운 훈련 샘플  $x_M$ 을 찾는다. Binary classification에 영향을 미치는 중요한 변수라면  $d(x_i, x_M)$ 은 크고  $d(x_i, x_H)$ 는 작아야 할 것이다. 따라서 각 변수에 따른 점수  $S$ 의 초기값을 0으로 두고  $S = S - d(x_i, x_H) + d(x_i, x_M)$ 으로 업데이트한다. 그렇다면 변수의 개수만큼 점수  $S$ 가 생성될 것이고, 그 값이 높은 순서대로 ranking을 할 수 있다.

### • 비지도 변수선택

Target이 없는 데이터에 대해서도 특정 목적에 따라 중요한 변수만을 남기고 싶을 수도 있다. 이럴 때 기준으로서 사용할 수 있는 몇 가지 지표를 소개한다. 우선 당연히 데이터의 분산이 크거나 넓은 영역에 흩뿌려진 상태라면 불필요한 가능성이 높다. 또한 데이터의 분포가 uniform distribution을 따르면 변수가 갖는 엔트로피가 높다고 한다. 또한, 어떠한 변수의 값들의 변동성과 비교해서 여러 가지 오차 막대 (표준편차 · 표준오차 · CI)가 작을수록 reliable한 변수이다.

### • Forward vs. Backward

종종 Forward selection이 Backward selection 보다 계산적으로 이득이라는 이유로 nested subset of variables을 찾을 때 더 자주 사용되곤 한다. 그러나 Backward selection의 지지자들은 전진선택에서는 아직 선택되지 못한 변수를 고려해서 판단할 수 없기 때문에 더 약한 subset이 찾아질 수밖에 없다고 주장한다. 논문에서는 아래 예시를 통해 후진 소거법이 전진 선택을 outperform 할 수 있는 상황이 존재함을 보여준다.



위 예시에서 marginal하게 보았을 때 두 class를 가장 잘 분류하는 변수는 세 번째 변수이다. (Bottom Right) 따라서 전진선택을 진행하면 가장 먼저 세 번째 변수가 선택될 것이다. 그 후에는 첫 번째 변수가 추가 되었을 때 두 번째 변수가 추가 되었을 때보다 성능이 좋으므로 결과적으로 첫 번째와 세 번째 변수가 선택될 것이다. 그러나 그 두 변수의 조합이 클래스 분류에서 best performance를 보이지 못한다. 실제로 가장 좋은 성능을 보이는 건 첫 번째와 두 번째 변수의 조합이기 때문이다.

그렇다면 후진 소거법에서는 어떻게 될까? 후진 소거법에서는 전체 변수의 조합에서 어떤 변수를 제거 했을 때 성능이 좋아지는 지를 확인하므로, 세 번째 변수를 먼저 제거하는 방식으로 best subset을 찾아낼 수 있다.

개인적인 견해를 추가해보자면 변수가 정말 많은 경우에 최대한 적은 변수만으로 나쁘지 않은 성능을 보이고 싶다면 forward selection을 쓰는 게 좋을 것 같고, 차원 크기와 computing capacity에 그닥 제한이 없다면 backward selection을 쓰는 게 좋아 보인다.